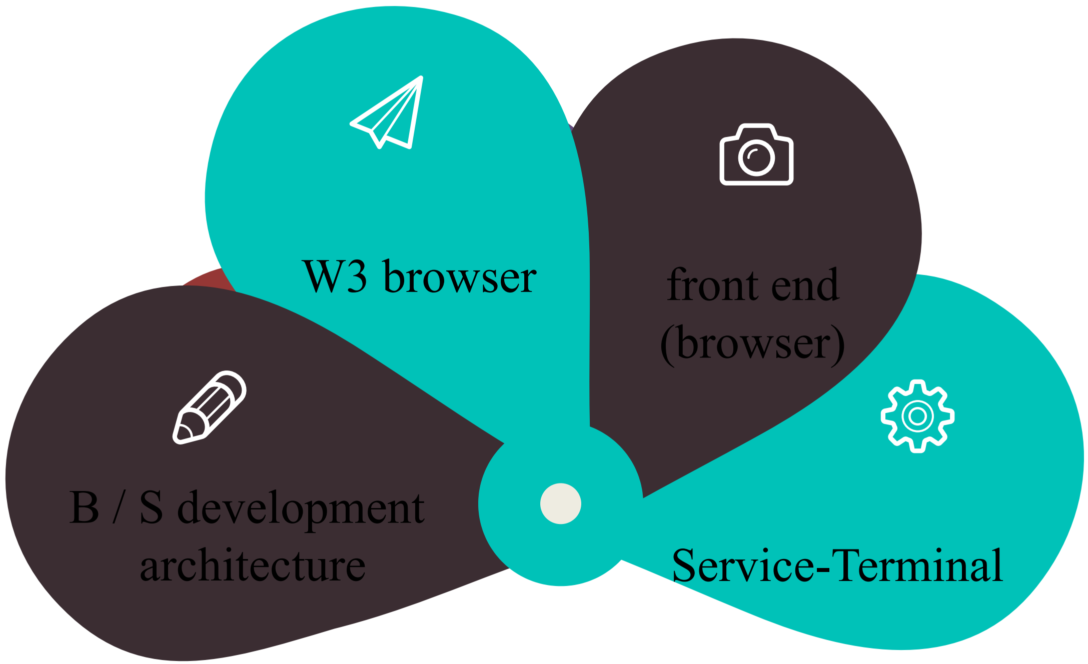
Big Data based Data Analysis Platform

Submitted by: Bikang Peng
ID: 5919394

SC6360 (Artificial Intelligence)
Instructor: Asst. Prof. Dr. Anilkumar K. Gopalakrishnan

Contents:

System Architecture Design

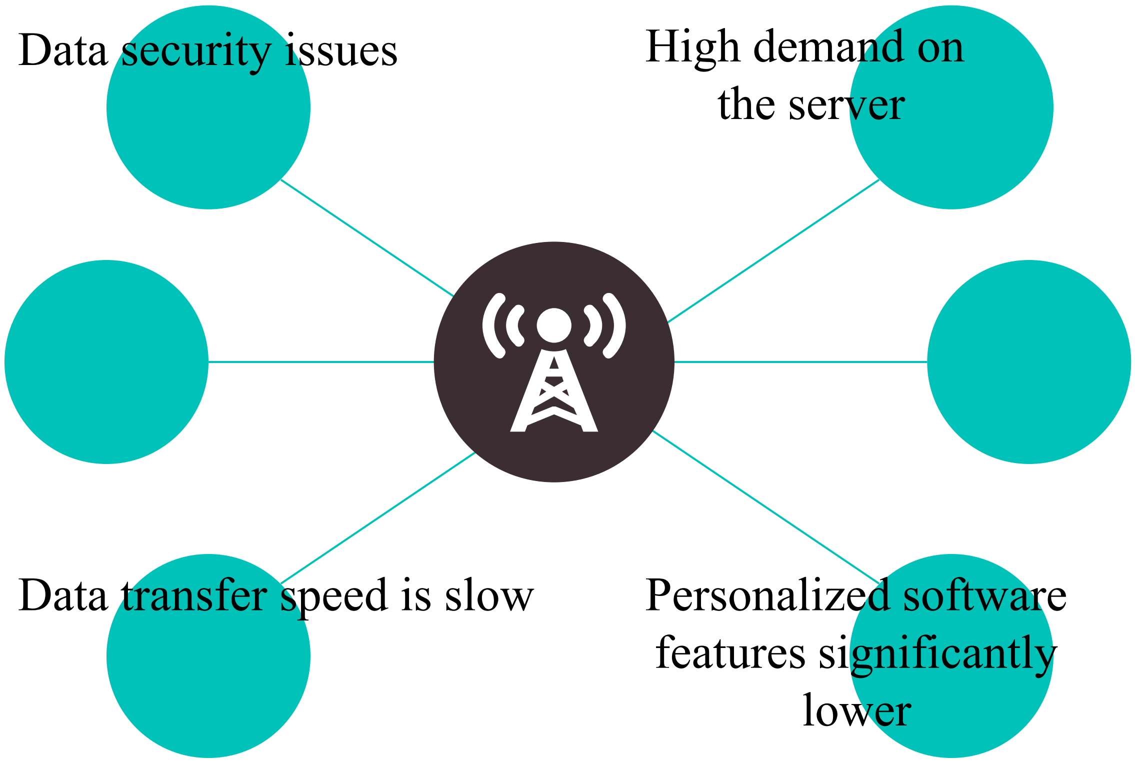


This greatly simplifies the client computer load (hence the name thin client), alleviating the system overhead of maintaining and upgrading, and lowering the overall cost of ownership (TCO) of the user.

The main features of BS

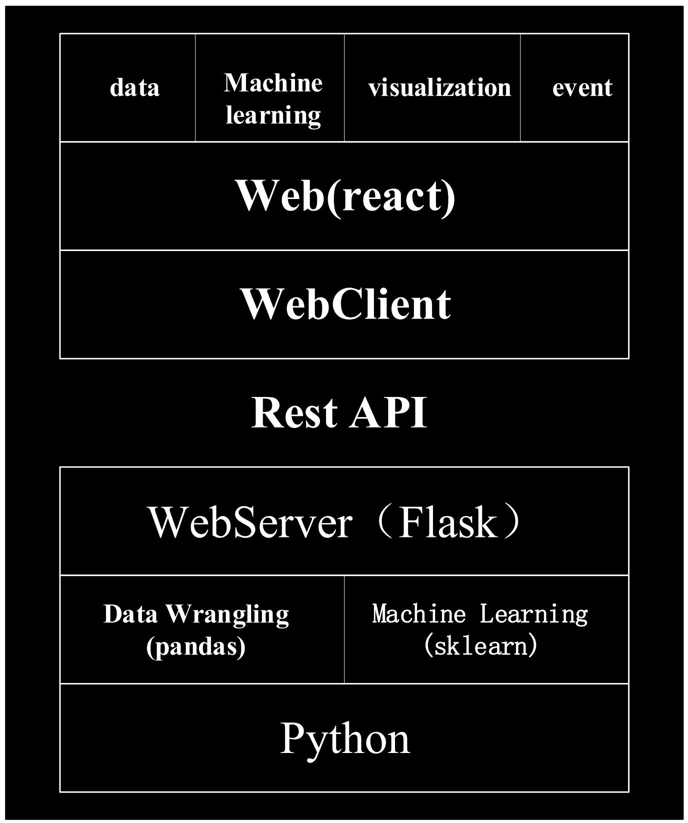
- Strong distribution
- Easy maintenance
- Easy to develop and share, low total cost of ownership

B / S deficiencies:



B / S summary

It is a thin client, for a large number of data entry and report replies, etc. need to interact with the browser through the browser, communication overhead, but also for the realization of complex application structure more difficult.

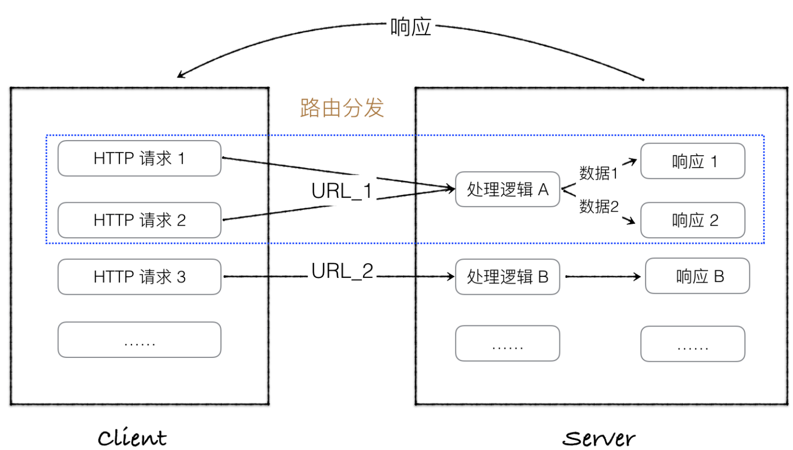


System Architecture

The server uses the flask framework

Flask is a lightweight web application framework written in Python. Based on Werkzeug WSGI Toolbox and Jinja2 Template Engine. Flask uses BSD license. Flask is also known as "microframework" because it uses a simple core and uses extensions to add other functionality. Flask does not use the default database, form validation tools.

However, Flask preserves the flexibility of amplification and can incorporate these capabilities with the Flask-extension: ORM, forms validation tools, file uploads, and a variety of open-source authentication technologies. Flask adopts the route distribution strategy, as shown in the following figure:



DATASETS

1. IRIS

Iris data sets are commonly used experimental data sets, collected by Fisher, 1936. Iris, also known as iris flower data set, is a type of multiple variable analysis data set. The dataset contains 150 datasets, divided into 3 categories, 50 for each category, and each containing 4 attributes. According to the four attributes of Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width, which one of the three species of Setosa, Versicolour, Virginica is predicted.

Attributes:

- Sepal.Length, the unit is cm, value Range: 0-8
- Sepal.Width, the unit is cm, value Range: 0-5
- Petal.Length, the unit is cm, value Range: 0-8
- Petal.Width, the unit is cm, value Range: 0-3

Species: Setosa, Versicolour, Virginica

DATASETS

2. Wine

Attributes:

1) Alcohol	value Range:0-15
2) Malic acid	value Range:0-6
3) Ash	value Range:0-4
4) Alkalinity of ash	value Range:0-30
5) Magnesium	value Range:0-200
6) Total phenols	value Range:0-4
7) Flavonoids	value Range:0-6
8) Non flavonoid phenols	value Range:0-0.8
9) Proanthocyanins	value Range:0-4
10)Color intensity	value Range:0-15
11)Hue	value Range:0-2
12)OD280/OD315	value Range:0-5
13)Proline	value Range:0-2000

Number of Instances

class 1 59   class 2 71   class 3 48

Algorithms

Classification

Clustering

Regression

Classification Algorithms

The classification algorithm uses three different classification learning algorithms, namely:

KNN(Main code)

```
from sklearn.neighbors import KNeighborsClassifier
from ml.classification.base import Classifier
class KNNClassifier(Classifier):
    def __init__(self):
        Classifier.__init__(self)
        self._name = "KNN"
        self._model = KNeighborsClassifier(n_neighbors=3)
```

Classification Algorithms

Bayes(Main code)

```
from sklearn.naive_bayes import GaussianNB
from ml.classification.base import Classifier
class NBayesClassifier(Classifier):
    def __init__(self):
        Classifier.__init__(self)
        self._name = "Bayes"
        self._model = GaussianNB()
```

SVM(Main code)

```
from sklearn import svm
from ml.classification.base import Classifier
class SVMClassifier(Classifier):
    def __init__(self):
        Classifier.__init__(self)
        self._name = "SVM"
        self._model = svm.SVC()
```

Classification Algorithms

The main code of the classification algorithm is:

```
def predictViz(self, scale):
    # Predict Viz only available for two dimensional dataset
    if len(self.  features[0]) != 2:
        return None
    result = dict()
    result["predict"] = list()
    result["data"] = list()
    # TODO leverage pandas to do this?
    range = dict()
    range["xmin"] = self.  features[0][0]
    range["xmax"] = self.  features[0][0]
    range["ymin"] = self.  features[0][1]
    range["ymax"] = self.  features[0][1]
    for item in self.  features:
        if item[0] > range["xmax"]:
            range["xmax"] = item[0]
        if item[0] < range["xmin"]:
            range["xmin"] = item[0]
        if item[1] > range["ymax"]:
            range["ymax"] = item[1]
        if item[1] < range["ymin"]:
            range["ymin"] = item[1]
    xstep = (float(range["xmax"]) - float(range["xmin"])) / scale
    ystep = (float(range["ymax"]) - float(range["ymin"])) / scale
    for x in xrange(0, scale):
        dx = range["xmin"] + x * xstep
        dy = range["ymin"]
        for y in xrange(0, scale):
            dy = dy + ystep
            onePredict = self.predict([[dx, dy]])
            record = dict()
            record["x"] = dx
            record["y"] = dy
            record["label"] = onePredict[0]
            result["predict"].append(record)
    for i in xrange(0, len(self.  label) - 1):
        record = dict()
        record["x"] = self.  features[i][0]
        record["y"] = self.  features[i][1]
        record["label"] = self.  label[i]
        result["data"].append(record)
    return result
```

Clustering algorithm

Clustering learning algorithm uses a K-means algorithm:

K-means(Main code)

```
from sklearn.cluster import KMeans
from ml.cluster.base import Cluster
class KMeansCluster(Cluster):
    def __init__(self):
        Cluster.__init__(self)
        self._name = "KMeans"
        self._model = KMeans(n_clusters=3)
    # train the model with given data set
    def getParameterDef(self):
        pass
    def setParameter(self, parameter):
        pass
```

Clustering algorithm

The main code of the Clustering algorithm is:

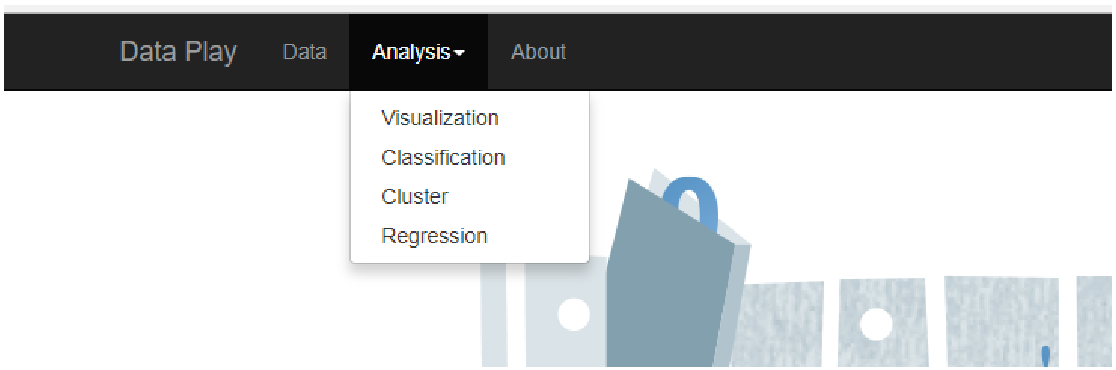
```
def predictViz(self, scale):
    # Predict Viz only available for one dimensional dataset
    if len(self._features[0]) < 2:
        return None
    result = dict()
    result["predict"] = list()
    result["data"] = list()
    predict_train = self.predict(self._features)
    for i in xrange(0, len(self._features)):
        item = dict()
        item["x"] = self._features[i][0]
        item["y"] = self._features[i][1]
        item["label"] = predict_train[i]
        result["data"].append(item)
    # TODO leverage pandas to do this?
    range = dict()
    range["xmin"] = self._features[0][0]
    range["xmax"] = self._features[0][0]
    range["ymin"] = self._features[0][1]
    range["ymax"] = self._features[0][1]
    for item in self._features:
        if item[0] > range["xmax"]:
            range["xmax"] = item[0]
        if item[0] < range["xmin"]:
            range["xmin"] = item[0]
        if item[1] > range["ymax"]:
            range["ymax"] = item[1]
        if item[1] < range["ymin"]:
            range["ymin"] = item[1]
    xstep = (float(range["xmax"]) - float(range["xmin"])) / scale
    ystep = (float(range["ymax"]) - float(range["ymin"])) / scale
    for x in xrange(0, scale):
        dx = range["xmin"] + x * xstep
        dy = range["ymin"]
        for y in xrange(0, scale):
            dy = dy + ystep
            onePredict = self.predict([[dx, dy]])
            record = dict()
            record["x"] = dx
            record["y"] = dy
            record["label"] = onePredict[0]
            result["predict"].append(record)
    return result
```

Regression Algorithm

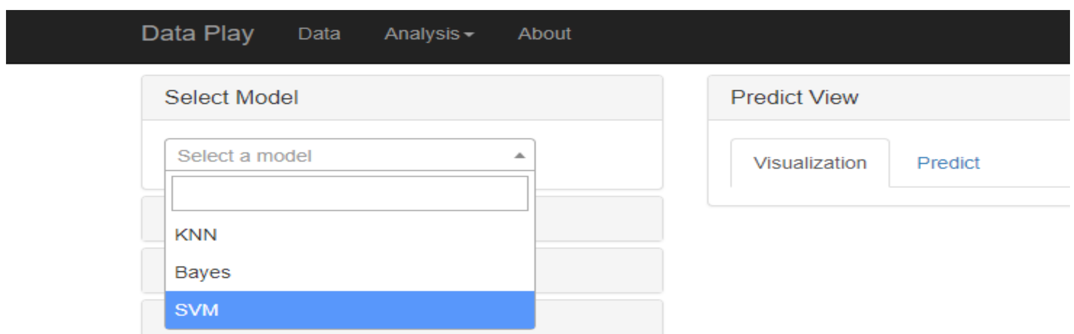
The regression algorithm uses two machine learning algorithms, linear and logistic, but the presentation is not yet complete.

Operation example

1) Select the type of analysis to the classification as an example

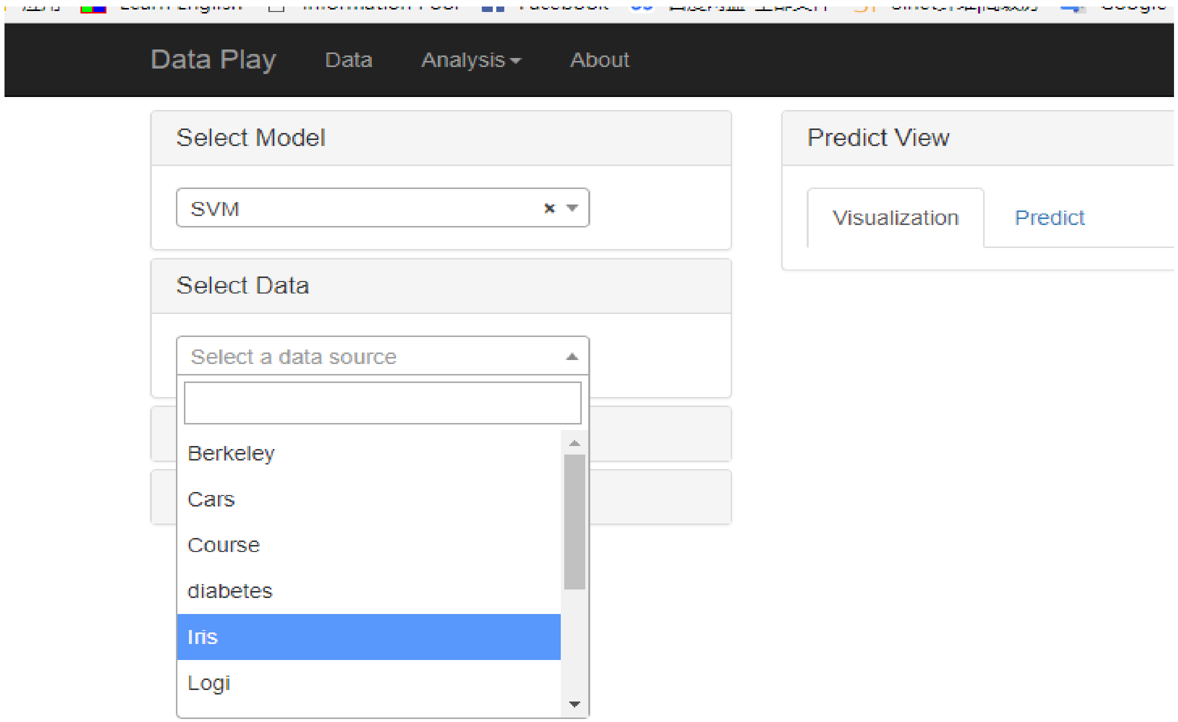


2) Select the model to the SVM as an example



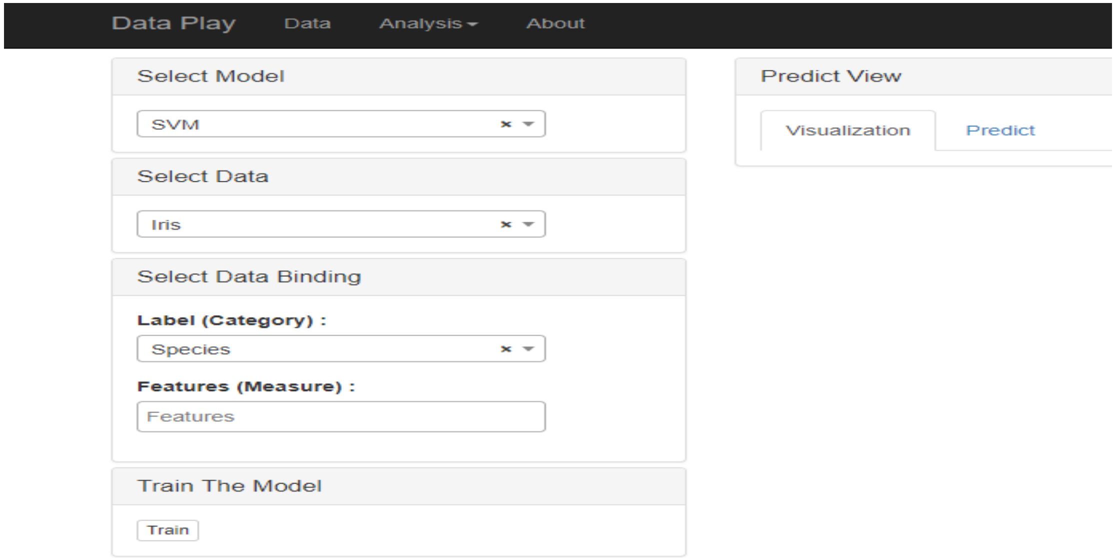
Operation example

3) Import data to iris as an example



Operation example

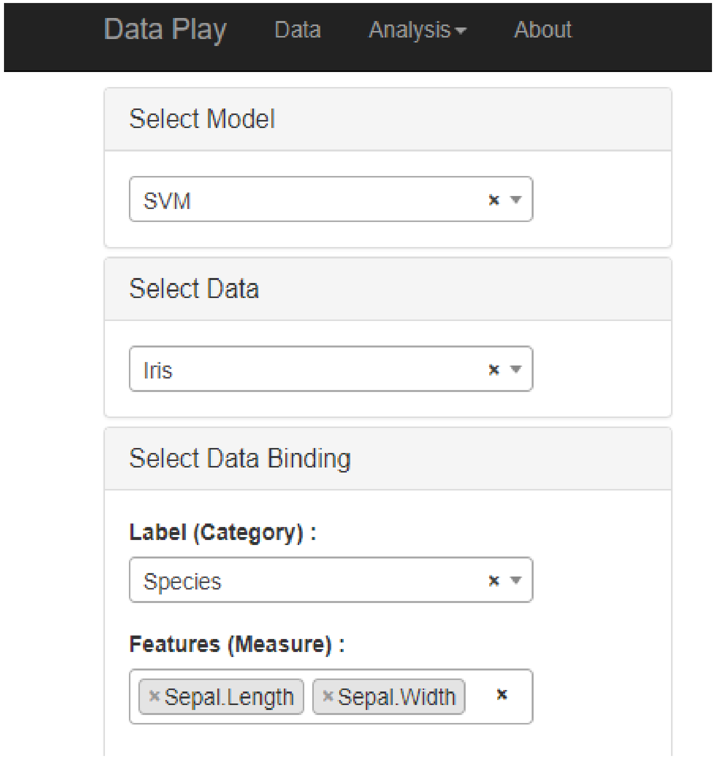
4) Select a data category: Species



Operation example

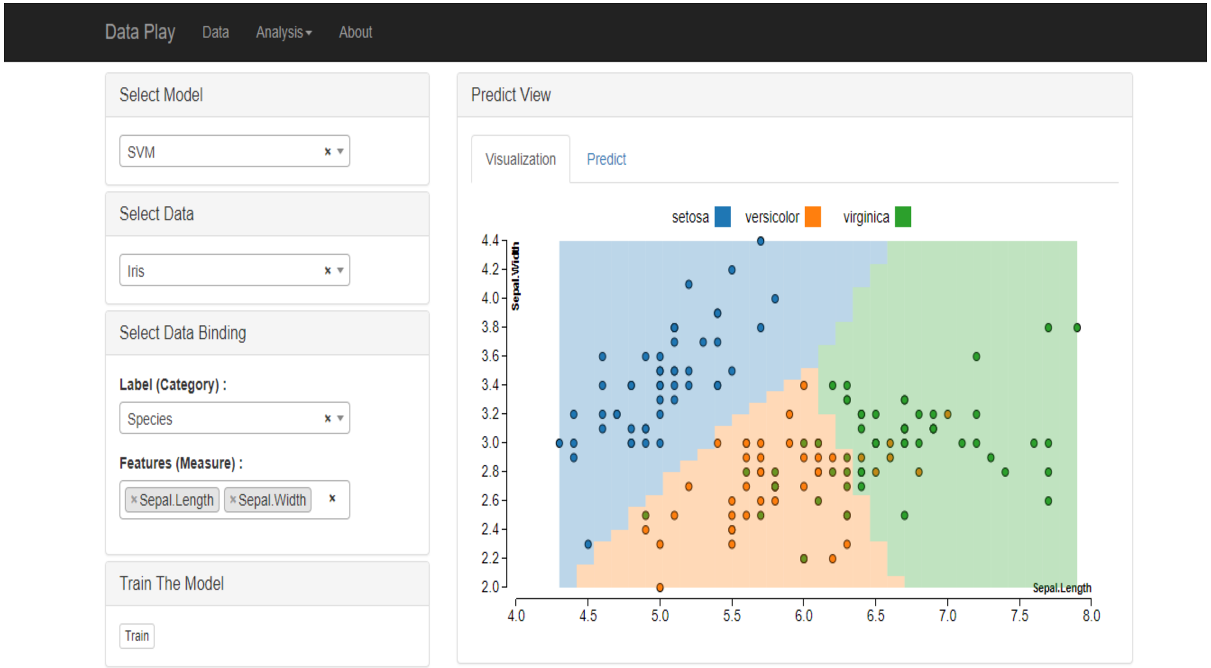
5) Select data Attributes: Can be 2 or more Attributes. But select 2 properties to see the visual interface, more than 2 properties can only be predicted can't see the visual interface, because the three-dimensional and higher dimensional two-dimensional display does not come out.

Take two Attributes as an example:



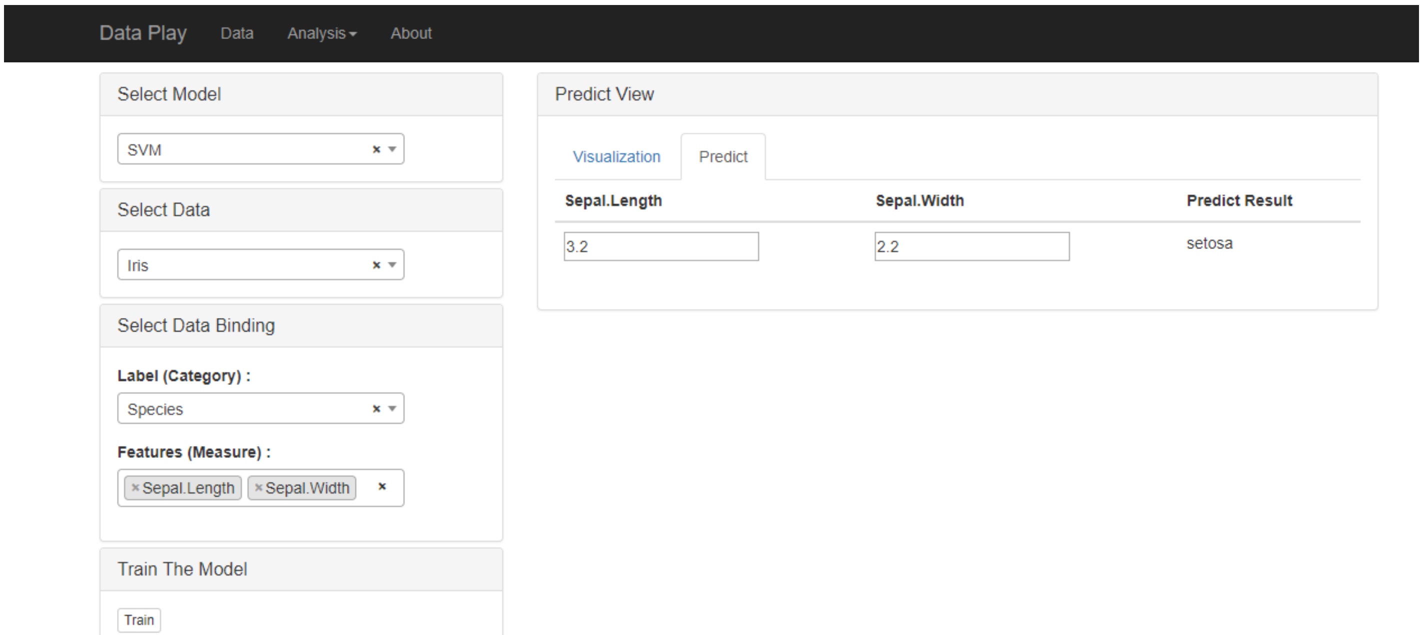
Operation example

6) Training, get visual interface



Operation example

7) Click predict, enter the data to predict, get the predict result



THANK YOU

FOR YOUR WATCHING